APPENDIX

S-PLUS CODE FOR POSITIONALLY CORRECTING ALGORITHM
version 1

```
"score.hts"<-
function(obj.hts, format = if(!is.null(fmt <- attr(obj.hts, "format"))) fmt
        else list(dim = c(8, 12), Rows = LETTERS[1:8], Columns = 2:11))
{
# obj.hts is an object of class "hts". This is a data.frame in which each
row
# represents one well of results and must contain the following NAMED
columns:
# REQUIRED
# $Plate: Plate id (alphanumeric factor) IN THE ORDER THE PLATES WERE
RUN
# $Row: Row id of the well (alphanumeric)
# $Col: Column id of the well (alphanumeric)
# $Type: Type of well contents:
# "D" for a sample compound or mixture
# "H" for high control (high raw measurements)
# "L" for low control (low raw measurements)
# ... other alphanumeric codes for other possible controls
# $Value: RAW measured value (NOT %inhibition or excitation)
#
#  OPTIONAL
# $Run: Runset code
# $Date: The Date on which the sample was run
# $Samp.ID: The sample ID code (e.g., L-number)
#
# NOTE: The data (plates) must be given in the order they were run
#
# More function arguments
# format: list with 3 components:
#  $dim = c(number of rows,number of columns)in plate
#  $Rows =  the row id's of test samples
# $Columns = the column id's of test samples
# The defaults given are for 96 well plates (where controls are in
# columns 1 and 12)
#
# Make sure Value column is numeric. If not, stop with error message.
        if(!is.numeric(obj.hts$Value)) stop("Value column must be numeric.")
        nrow <- length(format$Rows)
        ncol <- length(format$Columns)
        nplate <- nrow * ncol
        nm.obj <- names(obj.hts)
        # Make sure ordering of factors in data frame is maintained
        obj.hts$Plate <- ordered(obj.hts$Plate, levels =
unique(obj.hts$Plate))
        p.count <- table(obj.hts$Plate)
        bad.plt <- p.count > prod(format$dim)
        if(any(bad.plt)) {
                cat("\n\t\t ********** Bad Plate Indexing ***********\n\n
The following plate\n numbers appear more than once in the data:\n"
                        )
                bads <- p.count[bad.plt]
                cat("\tPLATE NUMB\t\tTotal Wells in Data with This Plate
Numb\n"
                        )
                bads <- paste("   ", names(bads), round(bads, 0), sep =
"\t")
                cat(bads, "\n", sep = "\n")
                stop()
        }
        platelist <- as.vector(unique(obj.hts$Plate))
        n.orig <- length(platelist)
        # Remove plates that are all controls, i.e. no sample wells("D") on
them
        good.plates <- as.vector(unique(obj.hts$Plate[obj.hts$Type == "D"]))
        lnth <- length(good.plates)
        if(lnth == n.orig)
                good.indx <- 1:n.orig
        else good.indx <- match(good.plates, platelist)
        plt.ind <- match(good.plates, obj.hts$Plate)
        #indices of good plates in plate column
        if(lnth < length(p.count)) {
                obj.hts <- obj.hts[!is.na(match(obj.hts$Plate,
good.plates)),
                        ]
                codes.new <- unique(codes(obj.hts$Plate))
                obj.hts$Plate <- structure(match(codes(obj.hts$Plate),
                    codes.new), levels =
levels(obj.hts$Plate)[codes.new],
                    class = c("ordered", "factor"))
        }
        pick.c <- c("Plate", "Row", "Col", "Value", "Samp.ID")
        if(is.na(match("Samp.ID", nm.obj)))
                samps <- obj.hts[obj.hts$Type == "D", pick.c[-5]]
        else samps <- obj.hts[obj.hts$Type == "D", pick.c]
        row <- match(samps$Row, format$Rows)
        col <- match(samps$Col, format$Columns)
        if(any(is.na(row)))
                stop("Row codes for sample wells does not match format
specification."
                    )
        if(any(is.na(col)))
                stop("Column codes for sample wells does not match format
specification."
                    )
        pl <- match(samps$Plate, good.plates)
        #   Fit an additive row/column fit for the samples on each plate
        y <- array(NA, c(nrow, ncol, lnth))
        samp.indx <- (pl - 1) * nplate + (col - 1) * nrow + row
        y[samp.indx] <- samps$Value
        fit.byplate <- apply(y, 3, function(x)
        twoway(x)[-4])
        # Make sure row and column effects haven't been corrupted by a row # or
column with a majority of actives by smoothing
        unl <- unlist(fit.byplate, rec = F, use.n = F)
        rowfits <- unlist(unl[seq(2, by = 3, length = lnth)])
        colfits <- unlist(unl[seq(3, by = 3, length = lnth)])
        grand <- unlist(unl[seq(1, by = 3, length = lnth)])
        na.smooth <- function(x, twice = T)
        {
# If length>=10, smooth
                if(sum(!is.na(x)) > 9) x[!is.na(x)] <- as.vector(smooth(x[!
                        is.na(x)], tw = twice))
                x
        }
        rowfits <- matrix(rowfits, ncol = nrow, byrow = T)
        colfits <- matrix(colfits, ncol = ncol, byrow = T)
        rowfits <- apply(rowfits, 2, na.smooth) # Smooth the row fits
        colfits <- apply(colfits, 2, na.smooth)     # Smooth the column fits
        # Create array of fits; layers = plates
        fit.a <- array(apply(cbind(rowfits, colfits), 1, function(x, nr, n)
        {
                outer(x[1:nr], x[(nr + 1):n], "+")
        }
        , nrow, nrow + ncol), dim = c(nrow, ncol, lnth)) + rep(grand, e =
                nplate)
        resid.a <- y - fit.a   #Residuals from smoothed row/column fits
####    Now smooth the residuals in each position over the Plate sequence
        if(lnth > 10) {
                resid.sm <- aperm(apply(resid.a, 1:2, na.smooth, twice = F),
c(
                        2, 3, 1))
                class(resid.sm) <- NULL
                fit.a <- fit.a + resid.sm       # final overall fits
                resid.a <- resid.a - resid.sm # final residuals
        }
        names(resid.a) <- NULL
########## Obtain the spread of the finals residuals on each plate as the mads
        mads <- apply(resid.a, 3, function(x, np)
        {
                mad(x, na = T) * sqrt(np/sum(!is.na(x)))   #df correction for missing
values
        }
        , nplate)
        # convert 0 mads to NA's to indicate that reasonable scores can't be
computed
        mads[mads == 0] <- NA
        # Compute means of low and high controls and activities
        act <- tapply(1:(dim(obj.hts)[1]), obj.hts$Plate, function(x, z, type)
        {
                z <- z[x]
                type <- type[x]
                lo <- mean(z[type == "L"], na.rm = T)
                hi <- mean(z[type == "H"], na.rm = T)
                diff <- hi - lo
                if(diff > 0)
                        act <- (100 * (z - lo))/diff
                else act <- rep(NA, length(z))
                list(lo, hi, act)
        }
        , obj.hts$Value, obj.hts$Type)
        unl <- unlist(act, rec = F, use.n = F)
        lows <- unlist(unl[seq(1, by = 3, length = n.orig)])[good.indx]
        highs <- unlist(unl[seq(2, by = 3, length = n.orig)])[good.indx]
        act <- unlist(unl[seq(3, by = 3, length = n.orig)])
        # Determine whether high or low controls are potent
        if(any(!is.na(lows)) && any(!is.na(highs))) {
                mlo <- median(lows, na.rm = T)
                mhi <- median(highs, na.rm = T)
                mgr <- median(grand, na.rm = T)
                if(abs(mlo - mgr) > abs(mhi - mgr))
                        potent <- "low"
                else potent <- "h"
        }
        else potent <- NA
        scores <- (resid.a/rep(mads, e = nplate))  # B-scores for all sample wells
        bsc <- rep(NA, e = dim(obj.hts)[1])
        bsc[obj.hts$Type == "D"] <- scores[samp.indx]
        # Housekeeping to track runs, number of plates per run, etc.
        if(is.na(match("Run", nm.obj)) |
length(unique(obj.hts$Run[plt.ind])) ==
                1) {
                Run <- lnth
                names(Run) <- "All"
                runset <- rep(1, lnth)
        }
        else {
                runset <- obj.hts$Run[plt.ind]
                Run <- table(runset)
        }
        if(!is.na(match("Date", nm.obj)))
                date <- as.character(obj.hts$Date[plt.ind])
        else date <- rep(NA, lnth)
        if(is.na(match("Samp.ID", nm.obj)))
                Samp.ID <- NA
        else {
                Samp.ID <- array(NA, c(nrow, ncol, lnth))
                Samp.ID[samp.indx] <- samps$Samp.ID
        }
        the.call <- sys.call()
        out <- structure(list(Call = the.call, Format = format, N.orig = n.orig,
                Potent = potent, Run = Run, Plate.stats = data.frame(Plate =
                good.plates, Date = date, Runset = runset, Center = grand,
                Scale = mads, Low.cntl = lows, Hi.cntl = highs, row.names = rep(
                NA, lnth), dup.row.names = T), Effects = list(Roweff = rowfits,
                Coleff = colfits), Results = list(Fitted = fit.a, Resid =
                resid.a, Samp.ID = Samp.ID), Activity = act, Bscore = bsc),
                class = "htsfit")
        if(all(is.na(out$Plate.stats$Low.cntl)) ||
all(is.na(out$Plate.stats$
                Hi.cntl)))
                class(out) <- c("bscr.only", class(out))
        invisible(out)
}

**********************************************************
```

S-PLUS CODE FOR POSITIONALLY CORRECTING ALGORITHM
version 2

```
"score.hts"<-
function(obj.hts, format = if(!is.null(fmt <- attr(obj.hts, "format"))) fmt
        else list(dim = c(8, 12), Rows = LETTERS[1:8], Columns = 2:11),
plate,
       plate.span = if(prod(format$dim) == 96) list(effects = 15, resids =
11,
               mads = 15) else list(effects = 11, resids = 11, mads = 11))
{
# obj.hts is an object of class "hts". This is a data.frame in which each
row
# represents one well of results and must contain the following NAMED
columns:
# REQUIRED
# $Plate: Plate id (alphanumeric factor) IN THE ORDER THE PLATES WERE
RUN
# $Row: Row id of the well (alphanumeric)
# $Col: Column id of the well (alphanumeric)
# $Type: Type of well contents:
# "D" for a sample compound or mixture
# "H" for high control (high raw measurements)
# "L" for low control (low raw measurements)
# ... other alphanumeric codes for other possible controls
# $Value: RAW measured value (NOT %inhibition or excitation)
#
#  OPTIONAL
# $Run: Runset code
# $Date: The Date on which the sample was run
# $Samp.ID: The sample ID code (e.g., L-number)
#
# NOTE: The data (plates) must be given in the order they were run
#
# More function arguments
# format: list with 3 components:
#  $dim = c(number of rows,number of columns)in plate
#  $Rows =  the row id's of test samples
# $Columns = the column id's of test samples
# The defaults given are for 96 well plates (where controls are in
# columns 1 and 12)
#
# Make sure Value column is numeric. If not, stop with error message.
        if(!is.numeric(obj.hts$Value)) stop("Value column must be numeric.")
        nrow <- length(format$Rows)
        ncol <- length(format$Columns)
        nplate <- nrow * ncol
        nm.obj <- names(obj.hts)
        # Make sure ordering of factors in data frame is maintained
        obj.hts$Plate <- ordered(as.character(obj.hts$Plate), levels =
unique(
                obj.hts$Plate))
        p.count <- table(obj.hts$Plate)
        bad.plt <- p.count > prod(format$dim)
        if(any(bad.plt)) {
                bads <- p.count[bad.plt]
                bads <- paste("    ", names(bads), "    ", round(bads, 0),
sep
                        = "\t", collapse = "\n")
                stop(paste("\n\t\t ********** Bad Plate Indexing
***********\n  \n  The following plate numbers appear more than once in the
data:\n\n   \tPLATE NUMBER\tTotal Wells in Data with This Plate Number\n",
                        bads, sep = ""))
        }
        platelist <- as.vector(unique(obj.hts$Plate))
        n.orig <- length(platelist)
        # Remove plates that are all controls, i.e. no sample wells("D") on
them
        good.plates <- as.vector(unique(obj.hts$Plate[obj.hts$Type == "D"]))
        lnth <- length(good.plates)
        if(lnth == n.orig)
                good.indx <- 1:n.orig
        else good.indx <- match(good.plates, platelist)
        indexofsamps <- obj.hts$Type == "D"
        if(lnth < length(p.count)) {
                obj.hts <- obj.hts[!is.na(match(obj.hts$Plate,
good.plates)),
                        ]
                codes.new <- unique(codes(obj.hts$Plate))
                obj.hts$Plate <- structure(match(codes(obj.hts$Plate),
                        codes.new), levels =
levels(obj.hts$Plate)[codes.new],
                        class = c("ordered", "factor"))
        }
        plt.ind <- match(good.plates, obj.hts$Plate)
        #indices of good plates in plate column
        pick.c <- c("Plate", "Row", "Col", "Value", "Samp.ID")
        if(is.na(match("Samp.ID", nm.obj)))
                samps <- obj.hts[obj.hts$Type == "D", pick.c[-5]]
        else samps <- obj.hts[obj.hts$Type == "D", pick.c]
        row <- match(samps$Row, format$Rows)
        col <- match(samps$Col, format$Columns)
        if(any(is.na(row)))
                stop("Row codes for sample wells does not match format
specification."
                        )
        if(any(is.na(col)))
                stop("Column codes for sample wells does not match format
specification."
                        )
        pl <- match(samps$Plate, good.plates)
        #   Fit an additive row/column fit for the samples on each plate
        y <- array(NA, c(nrow, ncol, lnth))
        samp.indx <- (pl - 1) * nplate + (col - 1) * nrow + row
        y[samp.indx] <- samps$Value
        fit.byplate <- apply(y, 3, function(x)
        twoway(x, trim = 0.15)[-4])
        # Make sure row and column effects haven't been corrupted by a row
# or column with a majority of actives by smoothing
        unl <- unlist(fit.byplate, rec = F, use.n = F)
        rowfits <- unlist(unl[seq(2, by = 3, length = lnth)])
        colfits <- unlist(unl[seq(3, by = 3, length = lnth)])
        grand <- unlist(unl[seq(1, by = 3, length = lnth)])
        na.smooth <- function(x, span, delta = 2, method = "lowess")
        {
# The method= argument gives added resistance
# If length>span, smooth nonmissings
                ok <- !is.na(x)
                nok <- sum(ok)
                n <- length(x)
                if(nok > span) {
                        if(method == "tukey")
                                x[ok] <- as.vector(smooth(x[ok], twice = F))
                        span <- min(nok/2, span)
                        delta <- min(delta, 0.01 * nok)
                        x[ok] <- lowess((1:n)[ok], x[ok], f = span/nok,
delta
                                    = delta)$y
```

```
		}
		x
	}
	rowfits <- matrix(rowfits, ncol = nrow, byrow = T)
	colfits <- matrix(colfits, ncol = ncol, byrow = T)
	# Smooth the row fits
	spn <- plate.span$effects
	rowfits <- apply(rowfits, 2, na.smooth, span = spn)
	# Smooth the column fits
	colfits <- apply(colfits, 2, na.smooth, span = spn)
	# Create array of fits; layers = plates
	fit.a <- array(apply(cbind(rowfits, colfits), 1, function(x, nr, n)
	{
		outer(x[1:nr], x[(nr + 1):n], "+")
	}
	, nrow, nrow + ncol), dim = c(nrow, ncol, lnth)) + rep(grand, e =
		nplate)
	resid.a <- y - fit.a	#Residuals from smoothed row/column fits
#### Now smooth the residuals in each position over the Plate sequence
	if(lnth > 10) {
		resid.sm <- aperm(apply(resid.a, 1:2, na.smooth, span =
			plate.span$resids, method = "tukey"), c(2, 3, 1))
		class(resid.sm) <- NULL
		fit.a <- fit.a + resid.sm	# final overall fits
		resid.a <- resid.a - resid.sm	# final residuals
	}
	names(resid.a) <- NULL
	########## Obtain the spread of the finals residuals on each plate
as the mads
	mads <- apply(resid.a, 3, mad, na.rm = T)
	# convert 0 mads to NA's to indicate that reasonable scores can't be
computed
	mads[mads == 0] <- NA
	mads <- exp(na.smooth(log(mads), span = plate.span$mads))
	# Compute means of low and high controls and activities
	act <- tapply(1:(dim(obj.hts)[1]), obj.hts$Plate, function(x, z,
type)
	{
		z <- z[x]
		type <- type[x]
		lo <- mean(z[type == "L"], na.rm = T)
		hi <- mean(z[type == "H"], na.rm = T)
		diff <- hi - lo
		z <- z[type == "D"]
		if(diff > 0)
```

```
                act <- (100 * (z - lo))/diff
            else act <- rep(NA, length(z))
            list(lo, hi, act)
        }
        , obj.hts$Value, obj.hts$Type)
        unl <- unlist(act, rec = F, use.n = F)
        lows <- unlist(unl[seq(1, by = 3, length = lnth)])
        highs <- unlist(unl[seq(2, by = 3, length = lnth)])
        act <- unlist(unl[seq(3, by = 3, length = lnth)])
        # Determine whether high or low controls are potent
        if(any(!is.na(lows)) && any(!is.na(highs))) {
            mlo <- median(lows, na.rm = T)
            mhi <- median(highs, na.rm = T)
            mgr <- median(grand, na.rm = T)
            if(abs(mlo - mgr) > abs(mhi - mgr))
                potent <- "low"
            else potent <- "h"
        }
        else potent <- NA
        scores <- (resid.a/rep(mads, e = nplate))
        # B-scores for all sample wells
        bsc <- rep(NA, e = sum(p.count))
        Act <- bsc
        bsc[indexofsamps] <- scores[samp.indx]
        Act[indexofsamps] <- act
        # Housekeeping to track runs, number of plates per run, etc.
        if(is.na(match("Run", nm.obj)) |
length(unique(obj.hts$Run[plt.ind])) ==
            1) {
            Run <- lnth
            names(Run) <- "All"
            runset <- rep(1, lnth)
        }
        else {
            runset <- as.character(obj.hts$Run[plt.ind])
        ## use unique() to assure correct ordering in table()
            Run <- table(runset)[unique(runset)]
        }
        if(!is.na(match("Date", nm.obj)))
            date <- as.character(obj.hts$Date[plt.ind])
        else date <- rep(NA, lnth)
        if(is.na(match("Samp.ID", nm.obj)))
            Samp.ID <- NA
        else {
            Samp.ID <- array(NA, c(nrow, ncol, lnth))
```

```
                    Samp.ID[samp.indx] <- as.character(samps$Samp.ID)
          }
          the.call <- sys.call()
          out <- structure(list(Call = the.call, Format = format, N.orig =
n.orig,
                    Potent = potent, Run = Run, Plate.stats = data.frame(Plate =

                    good.plates, Date = date, Runset = runset, Center = grand,
                    Scale = mads, Low.cntl = lows, Hi.cntl = highs, row.names =
rep(
                    NA, lnth), dup.row.names = T), Effects = list(Roweff =
rowfits,
                    Coleff = colfits), Results = list(Fitted = fit.a, Resid =
                    resid.a, Samp.ID = Samp.ID), Activity = Act, Bscore = bsc),
                    class = "htsfit")
          if(all(is.na(out$Plate.stats$Low.cntl)) ||
all(is.na(out$Plate.stats$
                    Hi.cntl)))
                    class(out) <- c("bscr.only", class(out))
          invisible(out)
}
```